✅ Junior-Level Java Interview Questions – Professional Answers

---

1. **Name some characteristics of object-oriented programming languages.**

Object-oriented programming (OOP) is based on four key principles:

Encapsulation (binding data and methods into a single unit)

Abstraction (exposing only relevant data to the user)

Inheritance (enabling new classes to use features of existing classes)

Polymorphism (allowing objects to behave differently based on their data type)

---

2. **What are the access modifiers you know? What does each one do?**

Public: Accessible from any class.

Private: Accessible only within the class it's defined in.

Protected: Accessible within the same package and subclasses.

Default (no modifier): Accessible only within the same package.

---

3. **What is the difference between overriding and overloading a method in Java?**

Overriding: Redefining a method in a subclass with the same signature.

Overloading: Creating multiple methods with the same name but different parameters in the same class.

---

4. **What's the difference between an Interface and an abstract class?**

Interface: Can only contain abstract methods (until Java 8+ added default/static methods).

Abstract class: Can contain both abstract and concrete methods.

Use interface for capability, abstract class for base behavior sharing.

---

5. **Can an Interface extend another Interface?**

Yes. Interfaces can extend one or more interfaces, allowing the new interface to inherit abstract method signatures.

---

6. **What does the static word mean in Java?**

The static keyword means the field or method belongs to the class rather than instances of the class.

---

7. **Can a static method be overridden in Java?**

No. Static methods are bound at compile time and can't be overridden, but they can be hidden using another static method in the subclass.

---

8. **What is Polymorphism? What about Inheritance?**

Polymorphism: Ability of an object to take on many forms—typically via method overriding.

Inheritance: Mechanism where one class inherits the fields and methods of another class.

---

9. **Can a constructor be inherited?**

No. Constructors are not inherited but can be invoked from a subclass using super().

10. **Do objects get passed by reference or value in Java?**

Java is strictly pass-by-value. For objects, the value passed is the reference to the object, not the object itself.

---

11. **What's the difference between using == and .equals() on a string?**

== compares memory addresses (object references).

.equals() compares the actual content of strings.

---

12. **What are hashCode() and equals() used for?**

They're used in collections like HashMap and HashSet to check equality and determine object placement. Objects that are equal must have the same hashCode.

---

13. **What does the interface Serializable do? What about Parcelable in Android?**

Serializable: A marker interface to enable object serialization.

Parcelable: Android-specific, faster serialization mechanism for inter-process communication.

---

14. **Why are Array and ArrayList different? When would you use each?**

Array: Fixed size, holds primitives or objects.

ArrayList: Resizable, part of Java Collections Framework.

Use Array for performance-critical tasks, ArrayList for dynamic data.

---

15. What's the difference between an Integer and int?

Int: Primitive data type.

Integer: Wrapper class that allows null values and is used in collections.

---

16. What is a ThreadPool? Is it better than using several "simple" threads?

A ThreadPool manages a group of reusable threads. It's more efficient than creating new threads for each task, reducing overhead and improving performance.

---

17. What's the difference between local, instance, and class variables?

Local: Defined inside methods; accessible only there.

Instance: Non-static fields tied to object instances.

Class: Static fields shared across all instances.

✅ Mid-Level Java Interview Questions – Professional Answers

---

1. **What is reflection?**

Reflection is a feature in Java that allows inspection and modification of classes, interfaces, methods, and fields at runtime. It's commonly used for frameworks, dependency injection, and testing.

---

2. **What is dependency injection? Can you name a few libraries?**

Dependency Injection (DI) is a design pattern where dependencies are passed to a class instead of being created within it. Popular libraries include Spring, Guice, and Dagger (Android).

---

3. **What are strong, soft, and weak references in Java?**

Strong: Regular reference. Object not garbage collected while referenced.

Soft: Object collected only if memory is low.

Weak: Object collected in the next GC cycle, even if referenced.

---

4. **What does the keyword synchronized mean?**

Synchronized ensures that only one thread at a time can execute a block or method, preventing race conditions during concurrent access.

---

5. **Can you have "memory leaks" in Java?**

Yes. While Java has garbage collection, improper use of listeners, static references, or caches can prevent unused objects from being collected, causing memory leaks.

---

6. **Do you need to set references to null in Java/Android?**

Not usually. The garbage collector handles this. However, in Android (e.g., in onDestroy), setting references to null can help avoid memory leaks due to context retention.

---

7. **What does it mean that a String is immutable?**

Once created, a String's value can't be changed. Any modification results in a new String object. This improves security and memory efficiency (via string pooling).

---

8. **What are transient and volatile modifiers?**

Transient: Field won't be serialized.

Volatile: Field updates are visible across threads immediately.

---

9. **What is the finalize() method?**

Finalize() is called by the garbage collector before destroying an object. It's deprecated in Java 9+ and discouraged due to unpredictability and performance issues.

---

10. **How does try {} finally {} work?**

The finally block always executes after the try block (regardless of exceptions), making it ideal for closing resources like streams or files.

---

11. What is the difference between instantiation and initialization of an object?

Instantiation: Allocating memory for an object (via new).

Initialization: Setting initial values (e.g., via a constructor).

---

12. When is a static block run?

A static block runs once when the class is first loaded, before any object creation or static method call.

---

13. Why are Generics used in Java?

Generics allow type-safe code by enabling classes, interfaces, and methods to operate on typed parameters, reducing the need for casting and preventing runtime errors.

---

14. Can you mention the design patterns you know? Which ones do you use?

Common design patterns:

Singleton (ensure one instance)

Factory (create objects)

Observer (event handling)

Builder (construct complex objects)

Strategy (select algorithm at runtime)

I frequently use Singleton, Factory, and Observer in application development.

---

15. Can you mention some types of testing you know?

Unit Testing (e.g., Junit)

Integration Testing

UI Testing (e.g., Espresso for Android)

Mock Testing (e.g., Mockito)

Regression Testing

Performance Testing

✅ Senior-Level Java Interview Questions – Professional Answers

---

1. **How does Integer.parseInt() work?**

Integer.parseInt(String s) converts a string to a primitive int. Internally, it validates the string format and parses each character as a digit based on base 10. If the format is invalid, it throws a NumberFormatException.

---

2. **Do you know what is the “double-checked locking” problem?**

It’s a concurrency issue in multithreaded environments where multiple threads may initialize a singleton simultaneously. In older Java versions (pre-Java 5), volatile didn’t ensure visibility, leading to partially constructed objects being accessed.

---

3. **Do you know the difference between StringBuffer and StringBuilder?**

Both are mutable, but:

StringBuffer is thread-safe (synchronized).

StringBuilder is faster but not thread-safe.

Use StringBuilder when thread safety is not required.

---

4. **How is StringBuilder implemented to avoid the immutable string allocation problem?**

StringBuilder maintains a mutable character array internally. It dynamically resizes and modifies this array, avoiding creation of multiple intermediate string objects, improving performance.

---

5. **What does Class.forName() do?**

It loads a class dynamically at runtime using its fully qualified name and executes static initializers. It's often used in JDBC to load the database driver class.

---

6. **What is Autoboxing and Unboxing?**

Autoboxing: Automatically converting primitives to wrapper classes (int → Integer)

Unboxing: Converting wrapper objects back to primitives (Integer → int)

This happens implicitly in assignments or expressions.

---

7. **What's the difference between an Enumeration and an Iterator?**

Enumeration: Older, read-only, used in legacy collections (e.g., Vector).

Iterator: Modern, supports element removal (remove()), used in collections framework.

---

8. **What is the difference between fail-fast and fail-safe in Java?**

Fail-fast: Throws ConcurrentModificationException if collection is modified during iteration (e.g., ArrayList, HashMap).

Fail-safe: Doesn't throw exception; uses a clone (e.g., ConcurrentHashMap, CopyOnWriteArrayList).

---

9. **What is PermGen in Java?**

PermGen (Permanent Generation) was a memory area in JVM (removed in Java 8) that stored metadata like class definitions and interned Strings. It was replaced by Metaspace for better memory management.

---

10. **What is a Java PriorityQueue?**

A PriorityQueue is a queue where elements are ordered based on their natural ordering or a custom comparator. The head of the queue is the least element according to the comparator.

---

11. **Is performance influenced by using the same number in different types (int, double, float)?**

Yes. Operations on int are generally faster due to lower memory footprint and simpler processing. Float and double are more computationally intensive due to floating-point arithmetic.

---

12. **What is the Java Heap?**

The Java Heap is a memory area used for dynamic allocation of objects at runtime. The JVM's garbage collector manages it. It's divided into young, old (tenured), and sometimes metaspace areas.

---

13. What is a daemon thread?

A daemon thread is a low-priority thread that runs in the background (e.g., garbage collection). It does not prevent the JVM from exiting once all non-daemon threads finish.

---

14. Can a dead thread be restarted?

No. Once a thread completes execution or is terminated, it enters the "dead" state and cannot be restarted. You must create a new thread instance to run again.
